

# Hierarchical Bayesian Models with Factorization for Content-Based Recommendation

Lanbo Zhang, Yi Zhang  
 School of Engineering  
 UC Santa Cruz  
 Santa Cruz, CA, USA  
 {lanbo, yiz}@soe.ucsc.edu

## ABSTRACT

Most existing content-based filtering<sup>1</sup> approaches learn user profiles independently without capturing the similarity among users. Bayesian hierarchical models [42] learn user profiles jointly and have the advantage of being able to borrow discriminative information from other users through a Bayesian prior. However, the standard Bayesian hierarchical models assume all user profiles are generated from the same prior. Considering the diversity of user interests, this assumption could be improved by introducing more flexibility. Besides, most existing content-based filtering approaches implicitly assume that each user profile corresponds to exactly one user interest and fail to capture a user's multiple interests (information needs).

In this paper<sup>2</sup>, we present a flexible Bayesian hierarchical modeling approach to model both commonality and diversity among users as well as individual users' multiple interests. We propose two models each with different assumptions, and the proposed models are called Discriminative Factored Prior Models (DFPM). In our models, each user profile is modeled as a discriminative classifier with a factored model as its prior, and different factors contribute in different levels to each user profile. Compared with existing content-based filtering models, DFPM are interesting because they can 1) borrow discriminative criteria of other users while learning a particular user profile through the factored prior; 2) trade off well between diversity and commonality among users; and 3) handle the challenging classification situation where each class contains multiple concepts. The experimental results on a dataset collected from real users on digg.com show that our models significantly outperform the baseline models of L-2 regularized logistic regression and traditional Bayesian hierarchical model with logistic regression.

<sup>1</sup>“filtering” and “recommendation” are interchangeable in this paper

<sup>2</sup>A short version of this paper is [35]

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 200X ACM X-XXXXX-XX-X/XX/XX ...\$10.00.

## Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval

## General Terms

Algorithms, Experimentation

## Keywords

Recommender System, Personalized Recommendation, Information Filtering, Content-based Filtering, Bayesian Hierarchical Model, Factorization

## 1. INTRODUCTION

Personalized recommendation has achieved great success in industrial applications, such as *Amazon*, *Netflix*, *Google News*, etc. As an alternative to search engines, recommender systems actively present items (products, movies, news articles, books, images, etc.) that are likely of interest to the user.

There are two families of algorithms for personalized recommendation: collaborative filtering and content-based filtering [34]. In this paper, we will focus on the content-based filtering. Existing content-based filtering research is largely influenced by the Filtering Track organized by TREC, where the task is to identify documents relevant to a specific topic from a document stream. There are two major approaches to handle this task. One is to use traditional IR retrieval models including Boolean model, traditional probabilistic models[24], vector space model[25], language models[10], etc. These approaches are initially designed for ranking and need to be combined with threshold setting algorithms[41] when used for filtering. Another approach is to treat recommendation as a classification task, and thus many existing machine learning approaches such as Naive Bayes, Decision Tree, Logistic Regression, SVM, Neural Networks, etc. could be used. However, both the traditional IR models and the traditional machine learning models learn each user profile independently and do not make use of the commonality among users.

A major challenge for a recommender system is the cold-start problem[26]. The recommendation performance for new users (or infrequent users) is usually poor since the amount of training data from such users is small. This problem may badly hurt the development of a new recommender system since it means new users must endure the poor initial performance. There has been much research

in machine learning fields on improving classification accuracy when only little training data is available. For example, semi-supervised learning is a way to use both unlabeled and labeled data to achieve this goal[43]; another approach is to introduce domain knowledge[11]. However, these approaches are not specific for the recommendation task and thus may not fit the recommendation task well. Yu et al[32] and Zhang et al[42] introduced the Bayesian hierarchical modeling approach to *jointly* learn user profiles for content-based filtering. Based on the fact that different users may have similar interests, the Bayesian hierarchical models assume that all user profiles are sampled from a common Gaussian prior. The Bayesian hierarchical modeling approach helps alleviate the cold-start problem since it is able to borrow discriminative information from other users through the common prior when learning a particular user profile, especially for those users with little training data.

However, some users may have totally different interests, and requiring these users' profiles to follow the same prior distribution may negatively influence the learned profiles, thus we need to trade off better between the diversity and commonality of user profiles. Besides, almost all existing content-based filtering approaches cannot capture the multiple interests of a user. They implicitly assume each user profile only corresponds to a single interest, which does not fit the real scenarios in personalized recommendation, where a real user's interests may contain multiple concepts/topics. For example, a graduate student working on information retrieval may be interested in both IR research advancements and NBA news.

To better model the diversity and commonality of users and individual users' multiple interests, we propose a more flexible Bayesian hierarchical modeling approach for personalized content-based recommendation. The proposed models aim at borrowing discriminative information, modeling the diversity of users, and capturing the multiple interests of each user. We propose a parameter learning algorithm based on point estimation for the proposed models. We evaluate the proposed models and the parameter learning algorithm on a real recommendation dataset. The experimental results show that the proposed models can significantly outperform the state-of-the-art content-based filtering algorithms.

## 2. RELATED WORK

The central task of a recommender system is to find relevant items for individual users. The two basic families of recommendation approaches are collaborative filtering and content-based filtering. **Collaborative filtering** doesn't need the content of an item and serves one user by leveraging information from other users with similar tastes and preferences. With its increasingly successful application in industry (e.g., Amazon, Netflix, Google News), collaborative filtering has attracted many researchers' attention during past several years. The memory-based [31][6] and model-based [14][27] approaches have been used in collaborative filtering task. **Content-based filtering** has been researched by the IR community for many years, especially in TREC 4-11 [18][19][15] [16][17][21][22][23]. Content-based filtering is the family of algorithms that make use of the characteristics of items and aim to understand which characteristics of items a particular user may like according to the user's history. In the scenario of content-based filtering, the recom-

mender system maintains a profile/classifier<sup>3</sup> for each user, which is usually in the form of a number of features and their corresponding weights. The Rocchio algorithm[25], in which user profiles and documents are represented as term vectors, can gradually update user profiles when relevance feedback from users is available [37, 30, 36, 39, 38, 9]. Zhang et al proposed to combine Rocchio algorithm with logistic regression through a Bayesian prior [40]. This algorithm outperforms both Rocchio algorithm and logistic regression. Ault et al[5] used kNN and Rocchio for information filtering. Cancedda et al[8], Srikanth et al[28] and Lewis[20] used SVM for TREC filtering tasks. Stricker et al[29] used neural networks to handle the case of few relevant examples in filtering.

Among all content-based filtering approaches, the Bayesian hierarchical model has achieved the state-of-the-art performances[42, 32]. It is quite related to our work, so we briefly describe it here. The most common Bayesian hierarchical model is shown in Figure 1 (we will use **BHM** to denote this model in the rest of this paper). There are  $M$  users in total and each user has  $J_m$  training examples.  $\mathbf{w}_m$  is the profile of user  $m$ . All user profiles follow a Gaussian prior distribution with mean vector  $\mu$  and covariance matrix  $\Sigma$ .

The other related work is the factorization-based topic models. For examples, Probabilistic Latent Semantic Analysis (PLSA)[13] and Latent Dirichlet Allocation (LDA)[7] are two generative models that model documents as a mixture of latent topics. These models are very attractive theoretically. However, we are not aware of any empirical evidence of their effectiveness in competitive retrieval and filtering tasks. This is not surprising since most of the successful models in these tasks are discriminative models, such as Logistic Regression, SVM or Neural Networks, instead of generative models, such as Naive Bays. The discriminative factorization models in this paper are motivated by the theoretical attractiveness of topic models as well as the empirical success of discriminative models. Though our models also introduce latent factors, there are clear differences between our models and PLSA/LDA. Unlike PLSA and LDA, our models are discriminative models, and their goals are to learn user profiles which are in the form of discriminative functions. The entries of each factor in our models are not necessarily words, and could be any item features such as the time or authority of a document.

Another much related work is multi-task learning (aka transfer learning, learning to learn) in the machine learning community. Several methods have been proposed to learn related tasks together, focusing on capturing the commonality among the tasks through a shared representation or a shared prior. The Bayesian hierarchical modeling approach can be viewed as a multi-task learning framework based on a shared prior, and the models in [42, 32] and this paper are both special cases of this framework. Zhang et al[33] proposed a probabilistic model to support a set of latent variable models for different multi-task learning scenarios. Their model is very similar to ours, however, the existing paper was focusing on multi-task learning and used Reuters document classification task for evaluation, while our work emphasizes on the application of personalized recommendation, which has very different characteristics and challenges.

<sup>3</sup>From the machine learning point of view, a user profile is a classifier

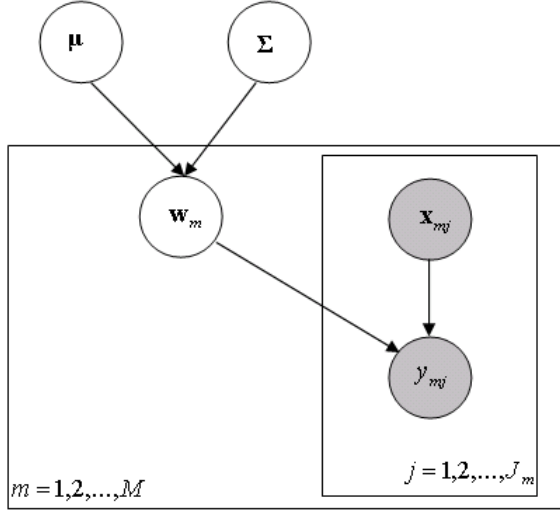


Figure 1: The Bayesian Hierarchical Model (BHM).  $\mu$  is the mean vector of the Gaussian prior, and  $\Sigma$  is the covariance matrix which is usually assumed diagonal. The shaded nodes denote observed variables.

### 3. DISCRIMINATIVE FACTORED PRIOR MODELS (DFPM)

The Discriminative Factored Prior Models (Figure 2) are motivated by the Bayesian hierarchical model in Figure 1.  $\Lambda$  is a  $K \times H$  matrix, and  $\lambda_m$  is a user-dependent vector with length  $H$ . The product of  $\Lambda$  and  $\lambda_m$  determines the prior mean of each user profile  $w_m$ , and  $\Sigma$  is the prior covariance matrix. The assumption of DFPM is: there are a number of hidden factors that represent different concepts in the item feature space; users may be interested in one or several of these hidden factors in different levels. Each column of matrix  $\Lambda$ , which is a  $K$ -dimensional vector, represents a specific hidden factor.  $\lambda_m$  tells how much each column of  $\Lambda$  contributes to the profile of user  $m$ .

To make our model description clear, we summarize the notations that will be used as follows:

- $H$ : the total number of hidden factors.
- $K$ : the total number of item/document features.
- $m = 1, 2, \dots, M$ : the index of individual users.  $M$  is the total number of users.
- $j = 1, 2, \dots, J_m$ : the index of training examples of user  $m$ .  $J_m$  is the number of training examples for user  $m$ .
- $w_m$ : the profile of user  $m$ , which is a  $K$ -dimensional vector.
- $\lambda_m$ : a user-related ( $m$ )  $H$ -dimensional vector that tells how much each hidden factor contributes to the user profile.
- $\langle x_{mj}, y_{mj} \rangle$ : the  $j$ -th training example of user  $m$ .  $x_{mj}$  is a  $K$ -dimensional vector representing a training item, and  $y_{mj} = 1/0$  means this item is relevant/irrelevant to the user.

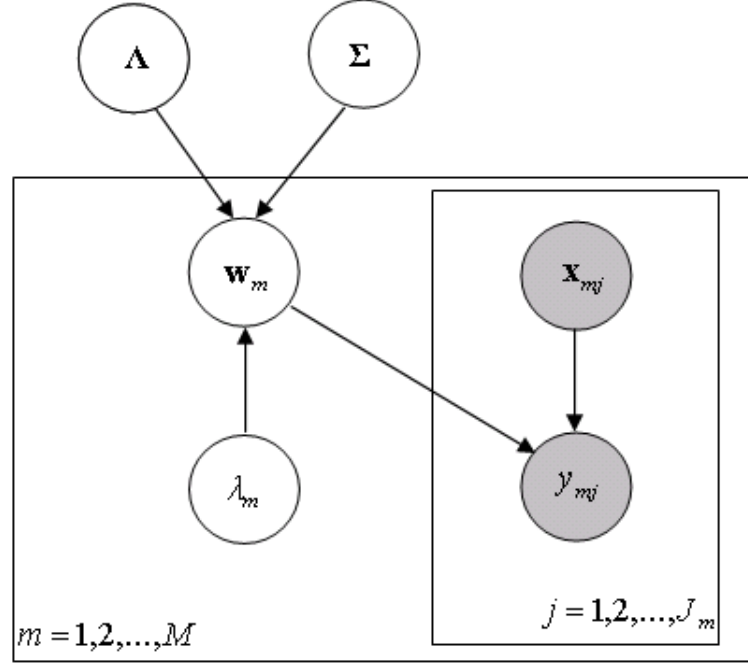


Figure 2: The Discriminative Factored Prior Model. The user profile  $w_m$  follows a normal distribution with mean  $\Lambda \lambda_m$  and variance  $\Sigma$ .  $\Lambda$  is a  $H$ -column matrix with each column representing a hidden factor. Each entry of vector  $\lambda_m$  reflects how much each hidden factor contributes to the user profile.

- $\Lambda$ : a  $K \times H$  matrix. Each column of  $\Lambda$  represents a hidden factor.
- $\Sigma$ : a  $K \times K$  covariance matrix.

$\lambda_m$  may follow two alternative distributions: Multinomial and Normal, and we will use **DFPM-Mult** and **DFPM-Norm** to denote these two models respectively. In DFPM-Mult, only one entry of  $\lambda_m$  is allowed to be 1 and all other entries be zero. This model clusters users into  $H$  groups, and users of the same group share a common hidden factor as the prior. We want to point out that the common Bayesian hierarchical model (BHM) is actually a special case of DFPM-Mult. When  $H = 1$ , DFPM-Mult is equivalent to BHM. In the case of DFPM-Norm,  $\lambda_m$  follows a Normal distribution with mean  $\mathbf{0}$  and variance  $b\mathbf{I}$ . DFPM-Norm assumes that each user may be interested in multiple hidden factors, and each entry of  $\lambda_m$  reflects how much the corresponding hidden factor is related to the user's interests. DFPM-Norm is used to capture each user's multiple interests.

We assume each user profile  $w_m$  is a random sample from a normal distribution with mean  $\Lambda \lambda_m$  and variance  $\Sigma$ . The label  $y_{mj}$  of a training item  $x_{mj}$  is  $y = f(x_{mj}; w_m)$ , where  $f$  could be many existing regression or classification models. We will take the logistic regression as an example to demonstrate how our models could be used for recommendation task. Let  $\mathbf{I}$  be an identity matrix,  $a, b, c$  be constant parameters, we summarize the discriminative factored prior models with logistic regression as follows:

- Each column of hidden factor matrix  $\Lambda$  follows a nor-

---

**Algorithm 1 : The Parameter Learning Algorithm**

---

- 1: Initialize  $\Lambda^0, i \leftarrow 0$
  - 2: Repeat:
  - 3: For each user  $m$ , compute  $\mathbf{w}_m^i, \lambda_m^i$  based on  $\Lambda^i$  by solving Equation 3
  - 4: Compute  $\Lambda^{i+1}$  based on  $\mathbf{w}_m^i, \lambda_m^i$  of each user according to Equation 10
  - 5:  $i \leftarrow i + 1$
  - 6: Until convergence
  - 7: Return  $\Lambda, \lambda_m, \mathbf{w}_m$
- 

mal distribution:  $N(\mathbf{0}, \mathbf{aI})$ .  $\Sigma$  follows an Inverse Wishart distribution  $\mathbf{W}^{-1}(\mathbf{I}, \mathbf{c})$

- The user vector  $\lambda_m$  may follow two alternative distributions: Normal or Multinomial. In the case of DFPM-Norm,  $\lambda_m \sim N(\mathbf{0}, \mathbf{bI})$ ; and in the case of DFPM-Mult,  $\lambda_m \sim \text{Multinomial}(\frac{1}{H}, \frac{1}{H}, \dots, \frac{1}{H})$
- The user profile  $\mathbf{w}_m$  follows a normal distribution:  $\mathbf{w}_m \sim N(\Lambda \lambda_m, \Sigma)$
- Given a user profile  $\mathbf{w}_m$  and an item feature vector  $\mathbf{x}_{mj}$ , its label is sampled from:

$$y_{mj} \sim \text{Bernoulli}\left(\frac{1}{1 + \exp(-\mathbf{w}_m^T \mathbf{x}_{mj})}\right)$$

Let  $\theta = (\Lambda, \Sigma, \lambda_m, \mathbf{w}_m)$  be the parameters of our models that need to be estimated. The joint likelihood of all variables and the observed data  $D = \{< \mathbf{x}_{mj}, y_{mj} >\}$  is:

$$P(D, \theta) = P(\Lambda)P(\Sigma) \prod_{m=1}^M P(\lambda_m)P(\mathbf{w}_m | \lambda_m, \Lambda, \Sigma) \prod_{j=1}^{J_m} P(y_{mj} | \mathbf{x}_{mj}, \mathbf{w}_m) \quad (1)$$

## 4. PARAMETER ESTIMATION

The empirical Bayes method[2] is often used when learning a complicated Bayesian hierarchical model like DFPM. However, the learning algorithm based on Empirical Bayes will be very complicated since there are many hidden variables  $(\Lambda, \Sigma, \lambda, \mathbf{w}_m)$  entangled in our models. Besides, the number of features and the number of users involved in the recommendation task are usually huge. Thus the empirical Bayes method is too computationally expensive to be used here. As an alternative, we present a simplified learning algorithm based on point estimation. This algorithm calculates a single value (the best guess) for each unknown variable.

### 4.1 The Learning Algorithm

To simplify the learning process, we assume the variance of  $\mathbf{w}_m$  is given, and as a result,  $\Sigma$  is removed from our models. Based on Equation 1, the maximum likelihood estimation of the parameters can be found based on Equation 2.<sup>4</sup> We introduce the constants  $c_1, c_2, c_3$  to replace the variance  $\Sigma, b, a$  respectively, and their model effects are equivalent.

<sup>4</sup>In this section, we use  $y_{mj} = -1$  to represent the label of a negative training example

---

**Algorithm 2 : The Step 3 of Model DFPM-Norm in Algorithm 1**

---

- 1: Initialize  $\lambda_m^{i,0}, h \leftarrow 0$
  - 2: Repeat:
  - 3: Compute  $\mathbf{w}_m^{i,h}$  based on  $\lambda_m^{i,h}$  by solving Equation 4
  - 4: Compute  $\lambda_m^{i,h+1}$  based on  $\mathbf{w}_m^{i,h}$  by Equation 7
  - 5:  $h \leftarrow h + 1$
  - 6: Until convergence
  - 7: Return  $\mathbf{w}_m^i, \lambda_m^i$
- 

---

**Algorithm 3 : The Greedy Algorithm for Step 3 of Model DFPM-Mult in Algorithm 1**

---

- 1:  $c \leftarrow \arg \min_c \sum_{j=1}^{J_m} \log [1 + \exp(-y_{mj}(\Lambda_c^i)^T \mathbf{x}_{mj})]$
- 2: Let the  $c$ -th entry of  $\lambda_m^i$  be 1 and all else be 0
- 3: Compute  $\mathbf{w}_m^i$  based on  $\lambda_m^i$  by solving Equation 4 using gradient based method
- 4: Return  $\mathbf{w}_m^i, \lambda_m^i$

Note:  $\Lambda_c^i$  is the  $c$ -th column of matrix  $\Lambda^i$

---

$$\begin{aligned} (\hat{\Lambda}, \hat{\lambda}_m, \hat{\mathbf{w}}_m) = & \arg \min_{\Lambda, \lambda_m, \mathbf{w}_m} \left\{ \sum_{m=1}^M \left\{ \sum_{j=1}^{J_m} \log [1 + \exp(-y_{mj} \mathbf{w}_m^T \mathbf{x}_{mj})] \right. \right. \\ & \left. \left. + c_1 \|\mathbf{w}_m - \Lambda \lambda_m\|^2 + c_2 \|\lambda_m\|^2 \right\} + c_3 \|\Lambda\|^2 \right\} \end{aligned} \quad (2)$$

To solve Equation 2, we first initialize  $\Lambda^0$ , then update the parameters by following the iteration steps A and B below.  $i$  is the iteration number. This process is also summarized in Algorithm 1.

- **Step A: find  $\mathbf{w}_m^i$  and  $\lambda_m^i$  for each user based on  $\Lambda^i$ .**

Given  $\Lambda^i$ , the user profiles are independent from each other. Thus we can focus on each individual user and solve the following subproblem for each user:

$$\begin{aligned} (\mathbf{w}_m^i, \lambda_m^i) = \arg \min_{\mathbf{w}_m, \lambda_m} & \left\{ \sum_{j=1}^{J_m} \log [1 + \exp(-y_{mj} \mathbf{w}_m^T \mathbf{x}_{mj})] \right. \\ & \left. + c_1 \|\mathbf{w}_m - \Lambda^i \lambda_m\|^2 + c_2 \|\lambda_m\|^2 \right\} \end{aligned} \quad (3)$$

To solve Equation 3, in the case of DFPM-Norm, we first initialize  $\lambda_m^{i,0}$ , then follow the iteration steps 1 and 2 below to find the solution.  $h$  is the iteration number. This process is also summarized in Algorithm 2.

- *Step 1: given  $\lambda_m^{i,h}$ , calculate  $\mathbf{w}_m^{i,h}$*

$$\begin{aligned} \mathbf{w}_m^{i,h} = \arg \min_{\mathbf{w}_m} & \left\{ \sum_{j=1}^{J_m} \log [1 + \exp(-y_{mj} \mathbf{w}_m^T \mathbf{x}_{mj})] \right. \\ & \left. + c_1 \|\mathbf{w}_m - \Lambda^i \lambda_m^{i,h}\|^2 \right\} \end{aligned} \quad (4)$$

There is no close form solution for Equation 4, so we choose to use gradient based method such as conjugate gradient decent[12] to solve it. Let  $L$  be the loss function in Equation 4, we can derive the following derivative for  $\mathbf{w}_m$ :

$$\frac{\partial L}{\partial \mathbf{w}_m} = \sum_{j=1}^{J_m} \frac{-y_{mj} \mathbf{x}_{mj}}{1 + \exp(y_{mj} \mathbf{w}_m^T \mathbf{x}_{mj})} + 2c_1 (\mathbf{w}_m - \Lambda^i \lambda_m^{i,h}) \quad (5)$$

- *Step 2: calculate  $\lambda_m^{i,h+1}$  based on  $\mathbf{w}_m^{i,h}$*

$$\lambda_m^{i,h+1} = \arg \min_{\lambda_m} \left\{ c_1 \left\| \mathbf{w}_m^{i,h} - \Lambda^i \lambda_m \right\|^2 + c_2 \left\| \lambda_m \right\|^2 \right\} \quad (6)$$

By taking derivative of  $\lambda_m$  on the loss function of Equation 6 and let it equal 0, we can get the following close form solution for Equation 6:

$$\lambda_m^{i,h+1} = \left( \Lambda^{iT} \Lambda^i + c_2 \mathbf{I} \right)^{-1} \Lambda^{iT} \mathbf{w}_m^{i,h} \quad (7)$$

In the case of DFPM-Mult, there are  $H$  alternative values for  $\lambda_m$ . A straightforward approach is to try each possible value of  $\lambda_m$ , and use a gradient search method to solve Equation 4. The  $\lambda_m$  and  $\mathbf{w}_m$  that lead to the minimum loss function value could be found finally. However, this approach is very time-consuming in real applications, especially when  $H$  is huge. Instead, we use a greedy algorithm (Algorithm 3) to approximately solve Equation 3. The basic idea is to set  $\mathbf{w}_m = \Lambda^i \lambda_m$  and find  $\lambda_m$  that minimizes the loss function in Equation 3. With the  $\lambda_m$  found, we then use gradient search to find the best  $\mathbf{w}_m$  based on Equation 4. This method is less time-consuming since the gradient search is only run once.

- **Step B: find  $\Lambda^{i+1}$  based on  $\mathbf{w}_m^i$  and  $\lambda_m^i$  of all users.**

$$\Lambda^{i+1} = \arg \min_{\Lambda} \left\{ \sum_{m=1}^M c_1 \left\| \mathbf{w}_m^i - \Lambda \lambda_m^i \right\|^2 + c_3 \left\| \Lambda \right\|^2 \right\} \quad (8)$$

Different rows of  $\Lambda$  are independent with each other, thus Equation 8 could be solved row by row. Let  $\Lambda_r^{i+1}$  be the  $r$ -th row vector of  $\Lambda^{i+1}$ , and  $\mathbf{w}_{m,r}^i$  be the  $r$ -th element of  $\mathbf{w}_m^i$ , then for  $r = 1, 2, \dots, K$ ,

$$\Lambda_r^{i+1} = \arg \min_{\Lambda_r} \left\{ \sum_{m=1}^M c_1 \left( \mathbf{w}_{m,r}^i - \Lambda_r \lambda_m^i \right)^2 + c_3 \left\| \Lambda_r \right\|^2 \right\} \quad (9)$$

By taking derivative of  $\Lambda_r$  on the loss function of Equation 9, we get the following solution for Equation 9:

$$\Lambda_r^{i+1} = \left( \left( \sum_{m=1}^M \lambda_m^i \lambda_m^{iT} + \frac{c_3}{c_1} \mathbf{I} \right)^{-1} \sum_{m=1}^M \mathbf{w}_{m,r}^i \lambda_m^i \right)^T \quad (10)$$

## 4.2 Time Complexity

In each iteration, the major computation work comprises the following three parts:

- The gradient search process involves the computation of the loss function in Equation 4 and the derivative in Equation 5. This takes time  $O(M \ln(\bar{J} + H)K)$ ,

**Table 1: Statistics of the crawled Digg data**

Number of users	15,162
Number of news articles	91,088
Total number of diggs	3,809,196
Average number of diggs per user	251

where  $M$  is the number of users,  $l$  is the total iteration number in algorithm 2 (for DFPM-Norm) or 1 (for DFPM-Mult),  $n$  is the average steps of the chosen gradient search algorithm,  $\bar{J}$  is the average number of training examples for each user,  $H$  is the number of hidden factors, and  $K$  is the number of features. This part takes up most of the running time.

- For DFPM-Norm, the computation of Equation 7, whose time complexity<sup>5</sup> is  $O(Ml(H^3 + HK))$ . For DFPM-Mult, this step is not included.
- The computation of Equation 10. For DFPM-Norm, the time complexity is  $O(H^3 + MHK)$ , and for DFPM-Mult, since  $\lambda_m$  has only one non-zero entry, the matrix that needs to be inverted is diagonal, thus the time complexity is  $O(MHK)$ .

We conclude that the time complexity of our learning algorithm is linear to: the number of users, the number of training examples, and the number of features. Besides, DFPM-Norm and DFPM-Mult are respectively cubic and linear to the number of hidden factors.

## 5. EXPERIMENTAL METHODOLOGY

### 5.1 The Dataset

To evaluate the proposed modeling approach, we collected a data set from Digg.com[1]. Digg.com is a website for people to share web content including news, images, and videos. Users can digg items they are interested in to promote the items' ranking. Each item digg by a user is considered a positive data point (a relevant document) for the user. We collected the complete digg history of news articles of more than 15,000 users. The detailed statistics of our dataset is shown in Table 1.

Since Digg.com only has user digg history available on its website, we couldn't get those articles users read but didn't digg. In other words, we don't have real negative examples. To address this problem, we randomly choose equal number<sup>6</sup> of articles which are not digg by a user as the negative examples for this user. Considering the large percentage of user undugg articles, we expect most of the articles sampled in this way are irrelevant to this user's interests.

### 5.2 Experimental Details

We randomly divide each user's data (including both positive and negative examples) into three parts: training (80%), validation (10%), and test (10%). The validation data is used to tune the parameters of both our models ( $H, c_1, c_2, c_3$ )

<sup>5</sup>We assume the time complexity of the matrix inversion algorithm is in  $O(H^3)$ .

<sup>6</sup>We didn't make the positive and negative classes unbalanced since the unbalanced problem is not the focus of this paper.

and the baseline models. We use the words as features. Both the stop words and rare words (occurring in less than 50 articles) are removed from the feature set. As a result, there are 35,865 features. When calculating the feature values, we use the TF\*IDF scoring method.

**Precision, Recall, and Macro-F1** are used as the evaluation measures. The conjugate gradient decent algorithm implemented in [4] is used in our implementation.

### 5.3 Experimental Goals

Our experiments are designed to answer the following questions:

- How is the performance of our models compared with the state-of-the-art algorithms?
- Can our models learn meaningful hidden factors, and how does the number of hidden factors ( $H$ ) influence the performance?
- Which assumption about  $\lambda_m$  works better, Multinomial or Normal distribution?
- How is the efficiency (running time) of our models?

To answer the first question, we compare our models with the L-2 regularized logistic regression (**L2LR**) and the Bayesian hierarchical model (Figure 1) with logistic regression (**BHLR**) implemented in [3]. By comparing our models with L2LR, which learns each user profile independently, we can tell whether our models can borrow useful discriminative information from other users. By comparing our models with BHLR, which is a state-of-the-art algorithm for content-based filtering [42], we want to evaluate whether our models can perform better by introducing a factored prior.

To answer the second question, the performances with different numbers of hidden factors will be compared. We will analyze the learned hidden factors and see whether the dominating features of each hidden factor look reasonable.

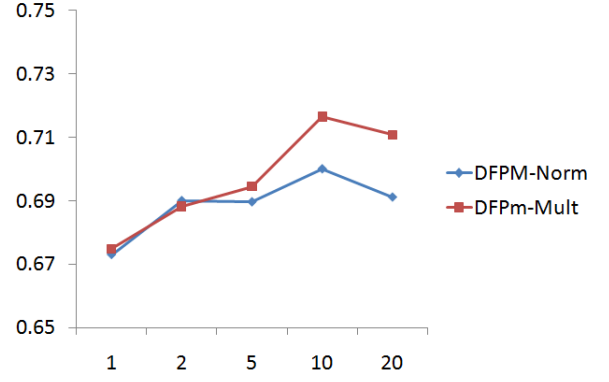
To answer the third question, we compare the two models corresponding to two assumptions of  $\lambda_m$ : the multinomial (**DFPM-Mult**) and normal (**DFPM-Norm**) distributions.

To answer the fourth question, we will record the running time of our learning algorithms with different numbers of users ( $M$ ) and different numbers of hidden factors ( $H$ ). We will see the overall running time of our algorithms and how the running time increases with the increase of the user number and the hidden factor number.

## 6. EXPERIMENTAL RESULTS

### 6.1 Overall Performances

The top-left graph in Figure 3 shows the overall performances of four algorithms. Both of our models (DFPM-Norm and DFPM-Mult) are statistically *significantly* better than the baselines in terms of Precision and Macro-F1 (based on t-test). The improvement on precision is very significant. This is very encouraging since Precision is a more important factor in most personalized recommender/filtering systems. To see whether our algorithms are helpful for both hard users (users with little training data) and easy users (users with much training data), we divide the users into five groups according to their numbers of diggs (less than



**Figure 4: Performances (Macro-F1) of our models at different  $H$  (number of hidden factors)**

50, 50-100, 100-200, 200-500, greater than 500 respectively) to see whether the performances for all kinds of users have been improved. The rest graphs in Figure 3 show the results on these user groups. We find our models outperform the baselines for all five user groups.

### 6.2 DFPM v.s. L2LR

Figure 3 shows that our models *significantly* outperform the L-2 regularized logistic regression, which learns each user profile independently. This demonstrates that our models successfully borrow discriminative information from other users by learning user profiles jointly.

### 6.3 DFPM v.s. BHLR

Figure 3 shows that our models *significantly* outperform the Bayesian hierarchical logistic regression model (BHLR). We find that BHLR already significantly outperforms L2LR, which indicates BHLR successfully borrows information from other users. Encouragingly, our models can further improve the performances. This demonstrates that our models can learn more accurate user profiles by introducing a factored prior.

Why our models can outperform BHLR? Not all users have similar interests, and it's not always a good idea to assume that all user profiles are generated from the same Gaussian distribution. Our models have less strong model assumptions and use the variable  $\lambda_m$  to model the diversity of users, and thus have the advantage of only borrowing information from *similar* users. In particular, users with similar interests share a common hidden factor as the prior in the DFPM-Mult model.

### 6.4 The Hidden Factors

Figure 4 shows how the number of hidden factors influences the performance. Remember BHLR is a special case ( $H = 1$ ) of our model DFPM-Mult. As  $H$  increases from 1 to 10, the performance keeps on improving and reaches the optimal value at  $H = 10$ . If we consider users with similar interests as a cluster, our model DFPM-Mult can effectively identify the underlying user clusters. To better understand the DFPM-Mult model, we list the most popular (dugg by most users) news articles of each user cluster in Table 2. It seems that the articles in each cluster somewhat represent similar topics. For example, the first cluster (Politics), the

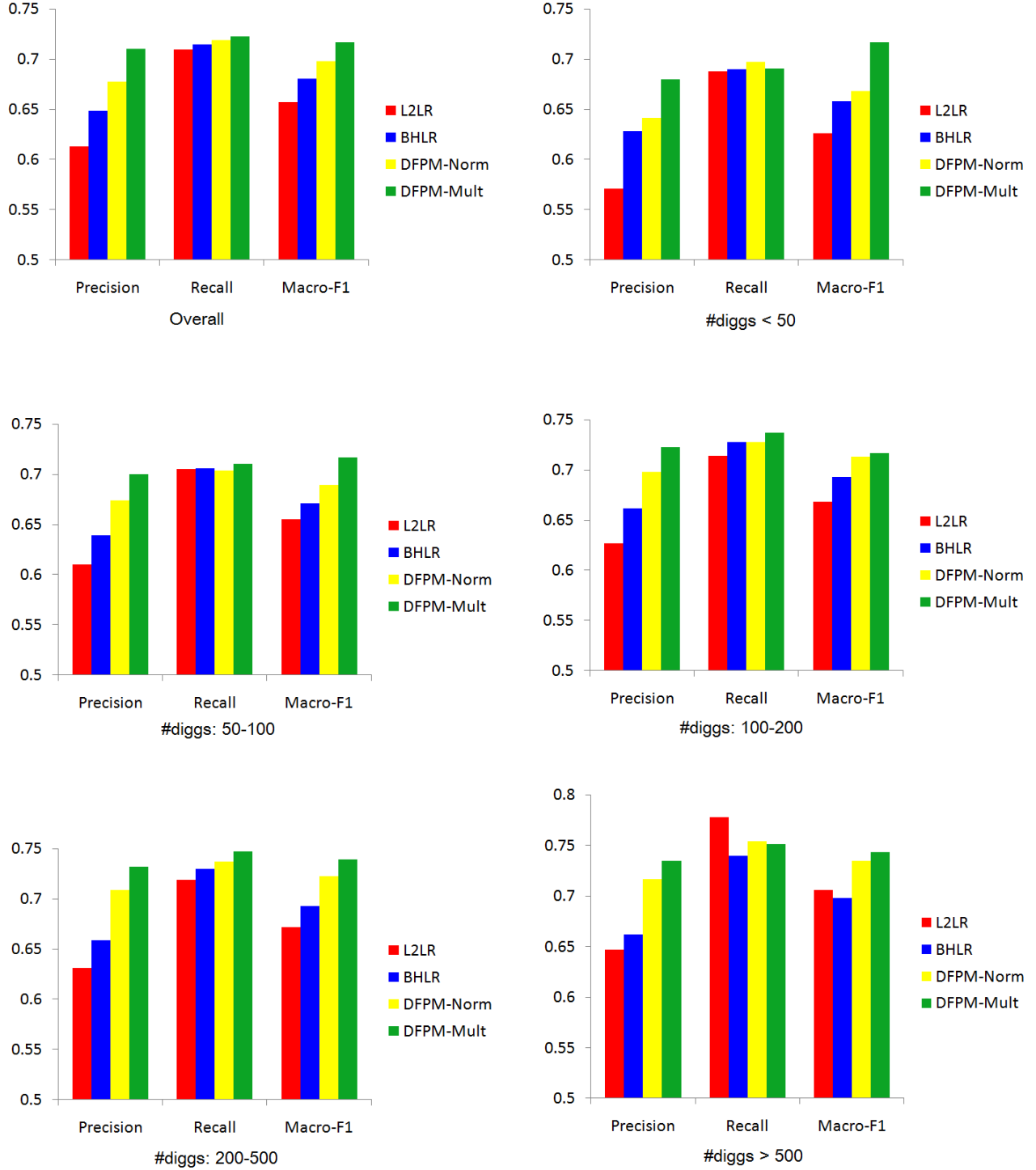


Figure 3: Performance comparison of different models. L2LR: L-2 regularized logistic regression. BHLR: Bayesian hierarchical logistic regression model. DFPM-Norm and DFPM-Mult are our models. The top-left graph shows the overall performance on all users, and the rest on five user groups with different #digs: less than 50, 50-100, 100-200, 200-500, greater than 500 respectively. According to our statistical test (t-test) results, our models *significantly* outperform the baseline methods.

**Table 3: Top words in some factors (by model DFPM-Mult).**

obama	scientist	smartphon	linux
presid	relationship	tablet	mozilla
jed	mlm	chrome	chrome
palin	exercis	android	diggTV
beck	geograph	dropbox	broadband
marijuana	treatment	feb	anonym
mccain	coach	broadband	techradar
yellow	copenhagen	diggTV	lifehack
barack	foreclosur	dialogg	dialogg
religi	orbit	chines	interfac
tortur	seti	tc	firefox
ko	npr	webcast	server
gop	emiss	idg	databas
pot	apprais	ie6	ie6
limbaugh	climat	anon	hacker
congress	techniqu	bing	odomzig
sarah	tablet	vulner	laptop
stewart	attract	hothardwar	pcworld
lewison	treehugg	infograph	rekcahefil
bailout	flight	aol	css

third (Apple products), and the fifth (Software). We also list the top 20 features (words) in some learned factors in Table 3. We observe that most of the words represent the concept of each hidden factor well.

### 6.5 $\lambda_m$ : Multinomial v.s. Normal

Figure 3 shows that DFPM-Mult performs better than DFPM-Norm. This is somewhat surprising. Initially, we expected that DFPM-Norm should perform better than DFPM-Mult since the Normal assumption of  $\lambda_m$  can capture the multiple interests of individual users. There are several possible reasons for this finding. First, it’s possible that the flexibility of  $\lambda_m$  makes the learning process more complicated. Second, the flexibility of  $\lambda_m$  may curtail the information borrowed from other users so that the commonality of similar users is not captured well. We are planning to investigate the reasons in more details in the future work.

### 6.6 The Running Time

Figure 5 shows the running performance of our learning algorithms. We report the running time (seconds) per iteration for both DFPM-Mult and DFPM-Norm with different numbers of hidden factors (left) and different fractions of users (right). The experiment was run on an Intel Xeon CPU (E5420@2.5GHz) with 16GB memory. The results in Figure 5 agree with the time complexity analysis in Section 4.2: 1) the time complexity of the learning algorithm for DFPM-Mult is linear to the number of users ( $M$ ) and the number of hidden factors ( $H$ ), and that for DFPM-Norm is linear to the number of users<sup>7</sup>; 2) DFPM-Mult is more efficient since it doesn’t include the inner iteration process in our learning algorithm (see section 4.1). Our algorithms converge reasonably fast and all the results reported in Fig-

<sup>7</sup>We didn’t find the time complexity of algorithm for DFPM-Norm is cubic to  $H$ , since when  $H$  is small, the running time is dominated by the gradient search steps in section 4.1

ure 3 are obtained within 20 iterations.

## 7. CONCLUSIONS

In this paper, we present a more flexible Bayesian hierarchical modeling approach for personalized content-based recommendation. Particularly, we propose two Discriminative Factored Prior Models and the corresponding parameter learning algorithms. Our modeling approach aims at: 1) learning user profiles jointly and borrowing discriminative criteria from other users by modeling the commonality of users through a shared factorized prior; 2) modeling the diversity of users and only borrowing information from *similar* users (DFPM-Mult); and 3) capturing multiple interests of individual users (DFPM-Norm).

We evaluate our models on a dataset collected from real web users on Digg.com[1], and compare them with two much related baseline algorithms. The experimental results demonstrate that:

- Our models significantly improve the recommendation performance, especially for users with little training data. Thus they can help alleviate the cold-start problem.
- It’s helpful to introduce a factorized prior. Particularly, the DFPM-Mult model learns more accurate user profiles since it can effectively cluster users with similar interests and has the advantage of only borrowing discriminative criteria from similar users while learning a particular user profile.
- The time complexity of our parameter learning algorithms is linear to the number of users. Thus our algorithms are applicable in real recommender systems.
- The multinomial assumption of  $\lambda_m$  seems to work better than the normal assumption.

In the future work, more research is needed to analyze the normal assumption of  $\lambda_m$ , since this model captures each user’s multiple interests and thus offers some advantages over the DFPM-Mult model. One possible approach is to try a Dirichlet prior instead of a normal prior for  $\lambda_m$ . Besides, we will also evaluate our models on more datasets, using explicit user feedback on irrelevant items. Our models can also be modified to fit the personalized recommendation task better, for example, to capture user interest drift by adding temporal variables.

## 8. REFERENCES

- [1] The digg website. <https://www.digg.com>.
- [2] Empirical bayes method. [http://en.wikipedia.org/wiki/Empirical\\_Bayes\\_method](http://en.wikipedia.org/wiki/Empirical_Bayes_method).
- [3] Hierarchical modeling in bbr. <http://www.stat.rutgers.edu/~madigan/BBR/hier.html>.
- [4] Macopt. <http://www.inference.phy.cam.ac.uk/mackay/c/macopt.html>.
- [5] T. Ault and Y. Yang. knn, rocchio and metrics for information filtering at trec-10. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, 2001.
- [6] R. M. Bell and Y. Koren. Improved neighborhood-based collaborative filtering.



Table 2: Top 5 most popular news articles of each user cluster (by model DFPM-Mult). For space limit, only the first five user clusters are shown.

Cluster NO.	#users in the cluster	Top 5 news articles
1	1235	Digg_This_If_You_Voted_For_Obama_2 Congress_Passes_Historic_Health_Care_Reform Barack_Obama_wins_the_2008_Presidential_Election Barack_Obama_Officialy_Becomes_44th_American_President We_had_eight_years_of_Bush_and_Cheney_Now_you_get_mad_2
2	255	Guns_Tumors_And_The_Limits_Of_The_Human_Eye High_Tech_Sport_Suits_May_Give_Skaters_Gold_Medal_Edge Lawmaker_Wants_to_Force_Kids_to_Wear_Cups_While_Bike_Riding New_Charging_Method_Could_Slash_Battery_Recharge_Times Deep_sea_pigs_used_to_investigate_dead_zones
3	874	Digg_The_Digg_iFrame_Toolbar_is_Dead_Unbanning_Domains 8_Things_That_Suck_About_the_iPad Google_to_offer_ultra_high_speed_broadband_in_US Apple_holding_iPhone_OS_4_event_April_8th Curious_user_opened_up_his_brand_new_iPad_to_see_PIC
4	1079	Callers_flood_ehealthinsurance_Where_s_My_Free_ObamaCare Teen_Sues_Mom_for_Hacking_Facebook_Account Childhood_Deafness_Gene_Uncovered Japan_s_whale_meat_obsession Skater_Jumps_Fails_Gets_Eaten_by_Shed
5	538	Ubuntu_9.10_Almost_Perfect Ubuntu_dumps_the_brown_introduces_new_theme_and_branding VLC_1.0.0_Released Linux_Users_Should_Say_Goodbye_Apple 10_skills_developers_will_need_in_the_next_five_years

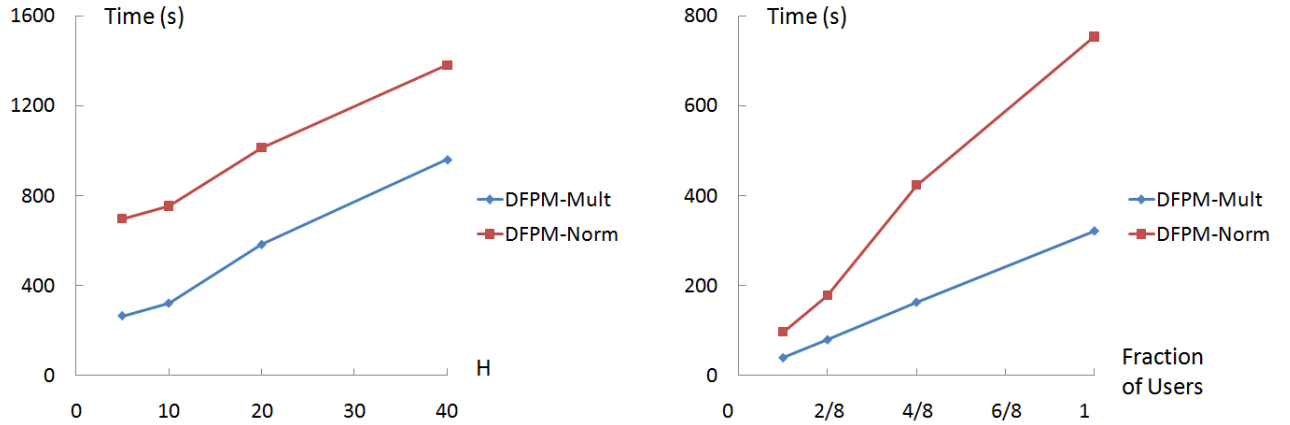


Figure 5: The running time (per iteration) of our learning algorithms. The left graph shows the running time (seconds) with different numbers of hidden factors ( $H$ ). The right graph shows the running time with different fractions of users (15,162 users in total).

- [7] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.
- [8] N. Cancedda, N. Cesa-Bianchi, A. Conconi, C. Gentile, C. Goutte, T. Graepel, Y. Li, J. M. Renders, J. S. Taylor, and A. Vinokourov. Kernel method for document filtering. In *Proceedings of the Eleventh Text REtrieval Conference (TREC-11)*, 2003.
- [9] Y. Chen, L. Zhang, A. Michelony, and Y. Zhang. 4is of social bully filtering: Identity, inference, influence, and intervention. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM '12*, pages 2677–2679, New York, NY, USA, 2012. ACM.
- [10] B. Croft, J. Lafferty, and editors. Language modeling for information retrieval. Kluwer, 2002.
- [11] A. Dayanik, D. D. Lewis, D. Madigan, V. Menkov, and A. Genkin. Constructing informative prior distributions from domain knowledge in text classification. In *SIGIR '06: Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 493–500, New York, NY, USA, 2006. ACM.
- [12] M. R. Hestenes and E. Stiefel. Methods of conjugate gradients for solving linear systems. *Journal of Research of the National Bureau of Standards*, 49:409–436, Dec 1952.
- [13] T. Hofmann. Probabilistic latent semantic analysis. In *Proceedings of Uncertainty in Artificial Intelligence, UAI'99*, pages 289–296, 1999.
- [14] T. Hofmann. Latent semantic models for collaborative filtering. *ACM Trans. Inf. Syst.*, 22(1):89–115, 2004.
- [15] D. A. Hull. The trec-6 filtering track: Description and analysis. In *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*, 1997.
- [16] D. A. Hull. The trec-7 filtering track: Description and analysis. In *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, 1998.
- [17] D. A. Hull and S. Robertson. The trec-8 filtering track final report. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, 1999.
- [18] D. D. Lewis. The trec-4 filtering track. In *Proceedings of the Fourth Text REtrieval Conference (TREC-4)*, 1995.
- [19] D. D. Lewis. The trec-5 filtering track. In *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*, 1996.
- [20] D. D. Lewis. Applying support vector machines to the trec-2001 batch filtering and routing tasks. In *Proceedings of the Eleventh Text REtrieval Conference (TREC-11)*, 2002.
- [21] S. Robertson and D. A. Hull. The trec-9 filtering track final report. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, 2000.
- [22] S. Robertson and I. Soboroff. The trec 2001 filtering track report. In *Proceedings of the Tenth Text REtrieval Conference (TREC-10)*, 2001.
- [23] S. Robertson and I. Soboroff. The trec 2002 filtering track report. In *Proceedings of the Eleventh Text REtrieval Conference (TREC-11)*, 2002.
- [24] S. Robertson and K. Sparck-Jones. Relevance weighting of search terms. In *Journal of the American Society for Information Science*, pages 129–146, 1976.
- [25] J. Rocchio. Relevance feedback in information retrieval. The SMART Retrieval System, 1971.
- [26] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 253–260, New York, NY, USA, 2002. ACM.
- [27] L. Si and R. Jin. Flexible mixture model for collaborative filtering. In *ICML '03: Proceedings of the Twentieth International Conference on Machine Learning*, 2003.
- [28] M. Srikanth, X. Wu, and R. Srihari. Ub at trec 11: Batch and adaptive filtering. In *Proceedings of the Eleventh Text REtrieval Conference (TREC-11)*, 2002.
- [29] M. Stricker, F. Vichot, G. Dreyfus, and F. Wolinski. Training context-sensitive neural networks with few relevant examples for the trec-9 routing. In *Proceedings of the Ninth Text REtrieval Conference (TREC-9)*, 2000.
- [30] Q. Xing, Y. Zhang, and L. Zhang. On bias problem in relevance feedback. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management, CIKM '11*, pages 1965–1968, New York, NY, USA, 2011. ACM.
- [31] K. Yu, A. Schwaighofer, V. Tresp, X. Xu, and H.-P. Kriegel. Probabilistic memory-based collaborative filtering. *IEEE Transactions on Knowledge and Data Engineering*, 16:56–69, 2004.
- [32] K. Yu, V. Tresp, and S. Yu. A nonparametric hierarchical bayesian framework for information filtering. In *SIGIR*, 2004.
- [33] J. Zhang, Z. Ghahramani, and Y. Yang. Flexible latent variable models for multi-task learning. *Mach. Learn.*, 73(3):221–242, 2008.
- [34] L. Zhang. *Content-based Filtering for Semi-structured Documents*. PhD thesis, Santa Cruz, CA, USA, 2013. AAI3609691.
- [35] L. Zhang and Y. Zhang. Discriminative factored prior models for personalized content-based recommendation. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management, CIKM '10*, pages 1569–1572, New York, NY, USA, 2010. ACM.
- [36] L. Zhang and Y. Zhang. Interactive retrieval based on faceted feedback. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '10*, pages 363–370, New York, NY, USA, 2010. ACM.
- [37] L. Zhang, Y. Zhang, J. d. Arma, and K. Yu. Ucsf at relevance feedback track. In *Proceedings of the 18th Text Retrieval Conference (TREC 2009)*, 2009.
- [38] L. Zhang, Y. Zhang, and Y. Chen. Summarizing highly structured documents for effective search interaction. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '12*, pages 145–154, New York, NY, USA, 2012. ACM.
- [39] L. Zhang, Y. Zhang, and Q. Xing. Filtering semi-structured documents based on faceted feedback.

In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 645–654, New York, NY, USA, 2011. ACM.

- [40] Y. Zhang. Using bayesian priors to combine classifiers for adaptive filtering. In *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 345–352, New York, NY, USA, 2004. ACM.
- [41] Y. Zhang and J. Callan. Maximum likelihood estimation for filtering thresholds. In *SIGIR '01*, pages 294–302, New York, NY, USA, 2001. ACM.
- [42] Y. Zhang and J. Koren. Efficient bayesian hierarchical user modeling for recommendation system. In *SIGIR '07*, pages 47–54, New York, NY, USA, 2007. ACM.
- [43] X. Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison, 2006.